Homework 5 ME333 - Winter 2025

Zhengyang Kris Weng Submission 02/03/2025

3. a. Referring to the Data Sheet, indicate which bits, 0-31, can be used as input/outputs for each of Ports A and B. For the PIC32MX795F512H in Figure 2.1, indicate which pin corresponds B0.

TRISA 15:0 for Port A TRISB 15:0 for Port B Pin 4 for B0

3. b. The SFR INTCON refers to “interrupt control.” Which bits, 0-31, of this SFR are unimplemented? Of the bits that are implemented, give the numbers of the bits and their names.

Bits 31:17, 15, 13, 11 and 7:5 are unimplemented.

Bit 16: SS0 Bit 14: FRZ Bit 12: MVEC Bit 10:8: TPC<2:0> Bit 4: INT4EP Bit 3: INT3EP Bit 2: INT2EP Bit 1: INT1EP Bit 0: INT0EP

4. Modify simplePIC.c so that both lights are on or off at the same time, instead of opposite each other. Turn in only the code that changed.\

See code `hw5q4.c` attached.

5. Modify simplePIC.c so that the function delay takes an int cycles as an argument. The for loop in delay executes cycles times, not a fixed value of 1,000,000. Then modify main so that the first time it calls delay , it passes a value equal to MAXCYCLES . The next time it calls delay with a value decreased by DELTACYCLES, and so on, until the value is less than zero, at which time it resets the value to MAXCYCLES . Use #define to define the constants MAXCYCLES as 1,000,000 and DELTACYCLES as 100,000. Turn in your code.

See code `hw5q5.c` attached.

7. The processor.o file linked with your simplePIC project is much larger than your final .hex file. Explain how that is possible

The .o file is large because it contains both the program’s code and a lot of extra information for debugging and linking, the final .hex file is much smaller because it only includes the essential instructions and data required for the device to operate.

9. Give three C commands, using TRISDSET, TRISDCLR, and TRISDINV, that set bits 2 and 3 of TRISD to 1, clear bits 1 and 5, and flip bits 0 and 4.

TRISDSET = 0b1100;
TRISDCLR = 0b100010; TRISDINV = 0b10001;

Chapter 4:

1. Identify which functions, constants, and global variables in NU32.c are private to NU32.c and which are meant to be used in other C files.

They're all public 😃

2. a. Remove the comments from invest.c in Appendix A. Now modify it to work on the NU32 using the NU32 library. You will need to replace all instances of printf and scanf with appropriate combinations of sprintf , sscanf , NU32_ReadUART3 and NU32_WriteUART3. Verify that you can provide data to the PIC32 with your keyboard and display the results on your computer screen. Turn in your code for all the files, with comments where you altered the input and output statements.

see code `hw5q2a.c` for more details.

2. b. Split invest.c into two C files, main.c and helper.c , and one header file, helper.h. helper.c contains all functions other than main . Which constants, function prototypes, data type definitions, etc., should go in each file? Build your project and verify that it works. For the safety of future helper library users, put an include guard in helper.h . Turn in your code and a separate paragraph justifying your choice for where to put the various definitions.

see codes in `hw5q2b` folder for more details.

`helper.h` contains function prototypes for the functions defined in `helper.c`. This allows `main.c` to use these functions. Whereas `helper.c` contains the actual definitions of the helper functions (getUserInput, calculateGrowth, sendOutput). This is where the logic for these operations resides. `main.c` is the entry point of the program.

2. c. Break invest.c into three files: main.c , io.c , and calculate.c. Any function which handles input or output should be in io.c . Think about which prototypes, data types, etc., are needed for each C file and come up with a good choice of a set of header files and how to include them. Again, for safety, use include guards in your header files. Verify that your code works. Turn in your code and a separate paragraph justifying your choice of header files.

see code in `hw5q2c` folder for more details.

The `main.c` file focuses on the overall control flow of the program. It initializes the system and invokes the appropriate functions to handle input, perform calculations, and output results. The `io.c` file contains the functions that handle input and output. Specifically, getUserInput prompts the user, parses the input, and checks its validity, while sendOutput handles printing the results to the user. The `calculate.c` file contains only the calculateGrowth function, which handles the core logic of calculating the growth of the investment over the specified number of years.